Available online at www.sciencedirect.com

ScienceDirect

Journal of Computer and System Sciences 74 (2008) 527–545

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.elsevier.com/locate/jcss

# Learning with belief levels ☆

Jānis Bārzdiņš, Rūsiņš Freivalds *, Carl H. Smith ✠

*Institute of Mathematics and Computer Science, University of Latvia, Raiņa bulvāris 29, LV-1459, Riga, Latvia*

Received 31 October 2004; received in revised form 17 May 2005

Available online 12 June 2007

**Abstract**

We study learning of predicate logics formulas from "elementary facts," i.e. from the values of the predicates in the given model. Several models of learning are considered, but most of our attention is paid to learning with belief levels. We propose an axiom system which describes what we consider to be a human scientist's natural behavior when trying to explore these elementary facts. It is proved that no such system can be complete. However we believe that our axiom system is "practically" complete. Theorems presented in the paper in some sense confirm our hypothesis.


*Keywords:* Learning; Inductive inference; Axiom systems; Completeness; Belief levels

## 1. Introduction

When E.M. Gold wrote his seminal paper [15], it was the starting point of a new area of research. Since then it was assumed that the main problem in the algorithmic learning theory is to restore a grammar from samples of the language, or a program from its sample computations. However scientists working in physics or biology have become accustomed to searching for interesting assertions rather than for a universal theory explaining everything. Following their example we consider in this paper how to find out nontrivial assertions from particular observations.

Two questions arise:

(1) What language can be used for these assertions.
(2) How to perform the learning process.

Since this is only one of the first papers on learning assertions from elementary facts, we restrict ourselves to the first-order predicate logics [18] as the language for the assertions. This language is rich enough for nontrivial assertions, and, on the other hand, it is also universal enough, it does not use specific properties of particular languages.

☆ This work was supported by an NSF International Award CCR 9421640.
* Corresponding author.
*E-mail addresses:* janis.barzdins@mii.lu.lv (J. Bārzdiņš), rusins.freivalds@mii.lu.lv (R. Freivalds).
✠ Deceased (1950–2004).

0022-0000/$ – see front matter 
doi:10.1016/j.jcss.2007.06.007

Learning formulas (of the first-order predicate logics) from elementary facts (values of the predicates in the given model) was started in [3–5,19].

The second question, how to perform the learning process, is a bit more complicated. A practitioner would demand finiteness of the learning process. The result would be produced after a finite number of computation steps. In contrast to this approach we are less interested in constructing practical learning algorithms, and we are more interested in understanding how such a learning process is performed by brain. Hence, we are ready to allow infinite learning process as well.

E.M. Gold [15] has already considered an infinite learning process when studying "identification of functions in the limit." We say that the inductive inference machine (IIM) identifies a function in the limit, if reading more and more input-output values $f(a) = b$, $f(c) = d, \ldots$, the IIM produces a sequence of hypotheses $p_0, p_1, p_2, \ldots$ about the program computing the function $f$, and this sequence has a limit being a correct program for $f$. Gold-style learning of formulas in the limit is also considered in our paper. However, our attention is concentrated to another type of infinite learning process.

In the paper [2] a new type of inductive inference "learning with confidence" (or "learning with belief levels") was considered. In some sense this notion is closer to the human inference process. When we produce any hypothesis based on a finite number of observations we are tended to estimate some belief level of our hypothesis. Thus, we come to a natural inductive inference type "learning with belief levels from elementary facts." This inductive inference type is central in our paper.

We start with considering how to learn recursive functions [22] with belief levels. After that we come to a natural inductive inference type "learning first-order predicate logic formulas with belief levels from elementary facts." By "elementary facts" we understand values of the given predicates for specific values of the arguments. Our main results concern axiomatization of learning formulas with belief levels.

Why do we believe in the great importance of the axiomatizability problem? The aim of axiomatization is to find the basic elements of which our practical everyday reasoning is constructed. At first it seems that inductive inference and axiomatization are incompatible notions because axiomatization traditionally is considered as a prerogative of deductive systems. However inductive inference processes (learning from examples) performed by distinct persons show great similarity. This prompts the existence of objective regularities in the inductive inference. On the other hand, axiomatization of inductive inference presumes search for objective regularities. It is evident that axioms of inductive inference should differ from axioms of deductive systems, because the latter serve for deducing absolute truth while the former are only hypotheses with higher or less high belief level.

Technically, axiomatization naturally introduces nondeterministic algorithms. In any axiom system there is never an algorithm which axiom is supposed to be used at the moment. This is always a nondeterministic choice.

We have discovered that nondeterminism, plus usage of belief levels may produce better computational power rather than nondeterminism, plus computation in the limit. We prove below (Theorems 5 and 6) that many more functions can be computed nondeterministically with belief levels rather than nondeterministically in the limit. Theorem 7 shows the same effect for learning functions. Theorem 15 shows a similar effect for learning predicate logic formulas.

Nevertheless, axiomatization is not merely a usage of nondeterministic algorithms. We would like to note that we have not been able to axiomatize the possibilities of identifiability in the limit. It seems that one of the reasons for this difficulty is the remoteness of identification type from the natural human inference. On the other hand, we have no doubt about the naturality of the learning with belief levels.

Finally, in Section 9 we discover that our axiom system is not complete. This may be considered as a serious defect. Yet, there are many famous incomplete axiom systems in mathematics. Remember the axioms of Euclidean geometry, axioms of first-order predicate logics (in these cases a complete axiomatization was achieved), axioms of Peano arithmetics, Zermelo–Fraenkel axiom system for the set theory. These systems are incomplete, and they cannot be made complete (by Gödel incompleteness theorem [13]). But this does not challenge the importance of this axiomatization. For instance, when we speak of Peano axiom system [20], then all *natural* assertions in arithmetics can be proved in Peano arithmetics (not always in a trivial way). On the other hand, all the counterexamples provided by the proof of the Gödel incompleteness theorem are *highly artificial*. The same can be said about David Hilbert's system of axioms of the elementary geometry [16] and about Zermelo–Fraenkel axiom system for the set theory [23].

We believe that our axiom system (being not complete) still is *kind of complete* in the following pragmatic sense. We hope that every true formula which can be deduced in a natural way from the infinite sequence of all elementary

facts in the given model by a human being, can be deduced by our axiom system as well. Of course, such an assertion cannot be proved formally.

## 2. Belief levels

In this section we consider algorithmic devices that compute by producing a sequence of numbers, each paired with a belief level. The definitions we give in this section will be used in subsequent sections to discuss computation combined with belief levels, learning with associated belief levels and a notion of proof joined with a belief level. This section contains the definitions and discussion that are shared by each of these three topics. In all cases we are concerned with processes that produce outputs. The interpretation of these outputs will determine whether a computation, a learning episode, or a proof is discussed.

A *belief level* is a rational number between 0 and 1. Intuitively, a 0 indicates no confidence in the answer and a 1 indicates certainty. As our confidence concerning the answer may vary over time, the processes we consider will produce a sequence of outputs, each paired with a belief level indicating the measure of certainty about the result. Such an output sequence of such pairs might look like: $y_0/e_0, y_1/e_1, y_2/e_2, \ldots$. We interpret these outputs as the value $y_i$ with the belief level $e_i$. The $y_i$'s are not necessarily distinct. In fact, for a given value $y$ there may be infinitely many $i$ such that $y = y_i$. In the case of a computation, the process producing the output sequence is algorithmic, and it is driven by a single input, e.g. the argument to the computation. Here the $y_i$'s in the output sequence are natural numbers from the range of the function being computed. For the learning case, the output producing process is again algorithmic, but it is driven by a potentially infinite input sequence, e.g. the examples, or a training set, for the learning algorithm.

The sequence $y_0/e_0, y_1/e_1, y_2/e_2, \ldots$ can be finite or infinite. We say that the sequence $y_0/e_0, y_1/e_1, y_2/e_2, \ldots$ converges to a value $e$ if either the sequence is finite and $e$ is the last member of the sequence, or the sequence is infinite and

$$e = \lim_{n\to\infty} e_n.$$

In general no monotonicity is demanded in the sequence $y_0/e_0, y_1/e_1, y_2/e_2, \ldots$. However in all proofs in this paper we use only monotone sequences of belief level, i.e. sequences such that $e_n \leqslant e_{n+1}$ for all $n$. Moreover, in all our results below the belief levels can be taken from a predetermined standard list, e.g.

$$e_n = 1 - \frac{1}{2^n}.$$

## 3. Computing with belief levels

This section contains our first application of belief level sequences. For the simplest case, we consider a traditional computation, augmented by self calculated belief levels. In this model the result of the computation becomes apparent only when the belief level for some potential result has reached 1. This event may not occur in any finite number of steps. A more natural name might be "limit computability," but there is already a well studied notion of that name. In fact, we show that our notion of computing with belief levels is different from limit computability. After defining the basic notions and giving some preliminary results we consider variations of our definition by relaxing the conditions for convergence of the belief sequence.

A function $g$ is called *limit computable* if there is a computable function $f$ such that, for any value $x$, $g(x) = \lim_{t\to\infty} f(x,t)$ [14]. By the *limit lemma* [7,14], the limit computable functions are precisely those functions $f$ that are Turing reducible to the halting problem.

### 3.1. Deterministic computing functions with belief levels

We start our discussion with definitions of analogues of computable relations and functions.

**Definition 1.** A relation $R$ is computable with belief levels if there is an algorithm that on any input $x$ produces an output sequence of pairs of a natural number $y$ and belief $s = y_0/e_0, y_1/e_1, y_2/e_2, \ldots$, such that the output sequence is

(nonstrictly) monotonically growing for every $y$ and $(x, y) \in R$ iff $E(s, y)$ iff the sequence of $y_0/e_0, y_1/e_1, y_2/e_2, \ldots$ corresponding to this $y$ converges to 1.

**Definition 2.** A function $f$ is computable with belief levels iff the relation $(x, f(x))$ is computable with belief levels.

**Theorem 1.** *For any total function $f$, if $f$ is limit computable, then $f$ is computable with belief levels.*

**Proof.** Suppose $f$ is limit computable. Then there is a recursive function $h$ such that for any $x$, $\lim_{t\to\infty} h(x, t) = f(x)$. The desired algorithm, on input $x$, outputs $h(x, t)$ with belief levels $1 - 1/2^t$ for $t = 0, 1, \ldots$. Let $s$ be the sequence of outputs so produced. The $E(s, f(x))$ monotonically approaches 1. Furthermore, for any $y \neq f(x)$ that is produced as a conjectured output by the algorithm, there is a largest $t$ such that $h(x, t) = y$. Hence, $E(s, y)$ converges to $1 - 1/2^t$. □

If the converse of Theorem 1 were also true, then the notion of recursiveness with belief levels would coincide precisely with the well studied notion of limit computability. However, as we will show, this is not the case. A consequence of the following discussion is that our new notion of computability with belief levels, despite several similarities, is distinct from the limit computability. In fact, by considering partial recursive functions, we will see that the two notions are incomparable. Our results rely on the notion of the graph of a function. For a partial function $\psi$, the graph of $\psi$ (denoted by $G_\psi$) is the set of ordered pairs $(x, \psi(x))$ as $x$ ranges over the domain of $\psi$. We continue with a characterization of computing with belief levels.

We use the arithmetical hierarchy of sets for this characterization. An excellent description of this hierarchy and of the analytical hierarchy of sets used in the next subsection can be found in [21].

By definition, a set $A$ is in $\Sigma_2$ if there is recursive, $\{0, 1\}$-valued function $g$ such that:

$$x \in A \quad \Leftrightarrow \quad \exists y \forall z \big[g(x, y, z) = 1\big].$$

It is convenient for us to use also an alternate characterization given by Kreisel, Shoenfield and Wang and presented as Theorem 4-XVIII in [21].

**Theorem 2.** *If $A \in \Sigma_2$, then there is a recursive $\{0, 1\}$-valued function $h$ such that*:

$$x \in A \quad \Leftrightarrow \quad \exists t' \forall t > t' \big[h(x, t) = 1\big],$$
$$x \notin A \quad \Leftrightarrow \quad \forall t' \exists t > t' \big[h(x, t) = 0\big].$$

By definition, a set $A$ is in $\Pi_2$ iff $\overline{A}$ is in $\Sigma_2$.

**Theorem 3.** *A function $\psi$ is computable with belief levels iff $G_\psi \in \Pi_2$.*

**Proof.** Let $\psi$ be an arbitrary partial recursive function. Recall that $G_\psi \in \Pi_2$ iff there is a total recursive function $f$ such that

$$(x, u) \in G_\phi \quad \Leftrightarrow \quad (\forall y)(\exists t)\big[f(x, u, y, t) = 1\big].$$

Suppose that $G_\psi$ is computable with belief levels via algorithm $A$. We will show the existence of a recursive function $f$ that will place $G_\psi \in \Pi_2$ via the above relation. Define:

$$f(x, u, y, t) = \begin{cases} 1, & \text{if } A \text{ on input } x \text{ outputs } u \text{ with belief level at least } 1 - 1/2^y \text{ in at most } t \text{ steps;} \\ 0, & \text{otherwise.} \end{cases}$$

Clearly, $f$ is recursive and it suffices to show that $G_\psi \in \Pi_2$. Suppose that $G_\psi \in \Pi_2$. Let $f$ be the recursive function of four arguments that is guaranteed to exist by the characterization of $\Pi_2$ above. To complete the proof, we must describe the operation of an algorithm that inductively computes $\psi$. This algorithm, on input $x$, will output $u$ with belief level $1 - 1/2^y$ if and only if a $t$ is found such that $f(x, u, y', t) = 1$ for all $y' \leqslant y$. The search for such a $t$ is accomplished via dovetailing. Hence, if $s$ is the sequence of outputs produced by the above algorithm on input $x$, then the belief level monotonically approaches 1 iff $\psi(x) = u$. □

Theorem 3 is easily contrasted with the following:

**Theorem 4.** *(See R. Freivalds and K. Podnieks [6], in English see [8].) A function $\psi$ is limit computable iff $G_\psi \in \Sigma_2$.*

*3.2. Nondeterministic computing functions with belief levels*

One of the traditional ways to relax the definition of computation is to consider nondeterministic computing agents. For completeness and comparison with our previous definition relaxation, we now consider the possibility to allow the algorithms that generate the belief sequences to be nondeterministic, giving rise nondeterministic computability with belief levels. Again, we will provide characterizations of what can be computed in terms of hierarchies based on traditional computation. To make the comparisons similar to those of the previous subsection, we use the analytical hierarchy. The analytical hierarchy is similar to the arithmetic hierarchy save for quantification over sets instead of integers [21]. Notationally, a superscript of 1 is used to differentiate the two hierarchies. Following the notation from [21] upper case letters are used for set variables and lower case letters for natural number valued variables. The main result about the analytical hierarchy that we use is given by the following lemma which is also known as exercise 16–10 from [21].

**Lemma 1.** *For any $n > 0$, any set of integers in $\Sigma_n^1$ can be expressed by a predicate with a prefix of $n + 2$ alternating quantifiers, starting with an existential quantifier, the first $n$ of which are over sets and the last two over natural numbers.*

For technical reasons, it is convenient to represent a nondeterministic machine as a deterministic one with a separate input tape for the nondeterministic choices. A run of a nondeterministic algorithm will be precisely determined by the input $x$ and the choice tape. Since any nondeterministic algorithm can make only finitely many choices at any given time instance, we can without loss of generality assume that there are only 2 possibilities to choose from at any moment. Hence, the choice tape uses an alphabet of $\{0, 1\}$. Associated with this tape is a set $S$ such that $i \in S$ iff a 1 appears in the $i$th square of the reserved input tape.

**Theorem 5.** *A function $\psi$ is nondeterministically computable with belief levels iff $G_\psi \in \Sigma_1^1$.*

**Proof.** Suppose that $\psi$ is nondeterministically computable with belief levels via a nondeterministic algorithm $A$. Let $s$ be the sequence of outputs produced by the computation of $A$ on input $x$ using choice set $S$. The sequence $E(s, u)$ monotonically converges to 1 iff $(x, u) \in G_\psi$. Let $P$ be the predicate that is true just in case the algorithm $A$ after $t$ steps on input $x$ with choice set $S$ has produced conjecture $u$ with belief level at least $1 - 1/2^y$. Notice that $P$ is recursive in $t$, $x$, $S$, $u$ and $y$. Hence,

$$(x, u) \in G_\psi \quad \Leftrightarrow \quad \exists S \forall y \exists t \big[P(t, x, S, u, y)\big]$$

is precisely a $\Sigma_1^1$ description of $G_\psi$. Suppose that $G_\psi \in \Sigma_1^1$. By Lemma 1, there is a recursive predicate $R$ such that

$$(x, u) \in G_\psi \quad \Leftrightarrow \quad \exists S \forall y \exists t \big[R(S, x, u, y, t)\big].$$

Using Theorem 2 this can be rewritten as

$$(x, u) \in G_\psi \quad \Leftrightarrow \quad \exists S \forall t' \exists t > t' \big[Q(S, x, u, y, t)\big]$$

for another recursive predicate $Q$. We now describe the behavior of a nondeterministic algorithm $A$ on input $x$ using a choice set $S$. $A$ operates in effective stages $t \geqslant 0$ described below.

**Begin stage $t$.** For each $u \leqslant t$, let $y$ be the cardinality of $\{t' \leqslant t \mid Q(S, x, u, y, t')\}$ and output $u$ with belief level $1 - 1/2^y$.
**End stage $t$.**

The proof is completed by the observation that for an arbitrary choice set $S$ for which $A$ produces output sequence $s$ on input $x$, $E(s, u)$ converges to 1 iff $\forall t' \exists t > t' Q(S, x, u, y, t)$. □

Theorem 5 contrasts nicely with

**Theorem 6.** *(See R. Freivalds and K. Podnieks [6], in English see [10].) A function $\psi$ is nondeterministically limit computable iff $\psi$ is limit computable iff $G_\psi \in \Sigma_2$.*

This comparison shows that nondeterminism provides a huge advantage of computation with belief levels over limit computation. The same type advantage is proved below for learning functions and learning formulas.

## 4. Learning functions with belief levels

The notion of computability with belief levels introduced in Section 3 has the character of producing the value of a function in the limit. In this sense it is reminiscent of *learning in the limit* where a correct program is determined in the limit based on examples of the intended input/output behavior. We review the basic notions of learning in the limit before proceeding to introduce an ostensibly new variant that incorporates the idea that one should be more confident of answer over time as the limit is approached.

Gold in his seminal paper [15], defined the notion called *identification in the limit*. This definition concerned learning by algorithmic devices now called *inductive inference machines* (IIMs). An IIM inputs the range of a recursive function, an ordered pair at a time and while doing so outputs computer programs. In this paper we will only discuss the inference of (total) recursive functions. We will assume, without loss of generality, that the input is received by an IIM in its natural domain increasing order, $f(0), f(1), \ldots$. An IIM, on input from a function $f$, will output a potentially infinite sequence of programs $p_0, p_1, \ldots$. The IIM *converges* if either the sequence is finite, say of length $n+1$, or there is a program $p$ such that for all but finitely many $i$, $p_i = p$. In the former case we say the IIM converges to $p_n$, and in the latter case, to $p$. In general, there is no effective way to tell when, and if, an IIM has converged.

When speaking about programs we are to remember that the identification power of IIMs depends very much on the programming language used, or equivalently, on the used numbering of partial recursive functions. Copying [9] and [11] we use the following definitions.

**Definition 3.** We say that $\{\Psi_i\}_{i=0}^{\infty}$ is a *computable numbering* of *PR* if there is 2-argument partial recursive function $U(n,x)$ such that:

(1) for arbitrary $n$, $\Psi_n(x) = U(n,x)$ for all $x$;
(2) for arbitrary 1-argument partial recursive function $\Psi$ there is a natural number $m$ such that $U(m,x) = \Psi(x)$ for all $x$.

**Definition 4.** We say that a numbering $\{\Psi_i\}_{i=0}^{\infty}$ of *PR* is reducible to the numbering $\{\Upsilon_i\}_{i=0}^{\infty}$ of *PR* if there is a total recursive function $f$ such that for arbitrary $i, x$ there holds $\Psi_i(x) = \Upsilon_{f(i)}(x)$.

**Definition 5.** We say that a computable numbering $\{\Upsilon_i\}_{i=0}^{\infty}$ of *PR* is a Gödel numbering if every computable $\{\Psi_i\}_{i=0}^{\infty}$ of *PR* is reducible to $\Upsilon_{f(i)}(x)$.

**Definition 6.** We say that a Gödel numbering $\{\Upsilon_i\}_{i=0}^{\infty}$ of *PR* is Friedberg numbering if for arbitrary natural numbers $i$ and $j$, $i \neq j$ implies $\Psi_i$ and $\Psi_j$ are distinct functions.

The existence of Friedberg numberings was proved in [12].

Following Gold we say that an IIM $M$ *identifies* a function $f$ if, when $M$ is given the graph of $f$ as input, it converges to a program $p$ that computes $f$. If an IIM identifies some function $f$, then some form of learning must have taken place, since, by the properties of convergence, only finitely much of the graph of $f$ was known by the IIM at the (unknown) point of convergence. Each IIM will learn some set of recursive functions. The collection of all such sets over the universe of effective algorithms viewed as IIMs serves as a characterization of the learning power inherent in the Gold model. This collection is symbolically denoted by EX (for explanation).

Now we will describe learning with belief levels. A *Belief inductive inference machine* (BIIM) works like an IIM, except that it outputs a sequence of hypothesis with belief levels: $s = p_0/e_0, p_1/e_1, p_2/e_2, \ldots$. We say that a BIIM $M$

learns a recursive function $f$ if on input from the graph of $f$ it outputs a sequence $p_0/e_0, p_1/e_1, p_2/e_2, \ldots$ such that for any $p_i$, the belief levels sequence is monotone and there exists a unique $p$ with the belief levels of $p$ approaching 1 and $\varphi_p = f$. In this way, each BIIM learns a set of recursive functions. The collection of all such sets is denoted by BEX.

**Theorem 7.** *The class $\Re$ of all 1-argument total recursive functions is BEX-identifiable.*

**Proof.** We will describe the operation of a BIIM $M$ as it tries to learn a recursive function $f$. The basic idea is for $M$ to run an enumeration technique [1,15] on a Friedberg numbering [12] of the partial recursive functions. More formally, let $\psi_0, \psi_1, \ldots$ be a Friedberg numbering, i.e. a listing of all and only the partial recursive functions such that if $i \neq j$ then $\psi_i \neq \psi_j$. Then there is a recursive function $h$ mapping the Friedberg indices to standard indices. Formally, $\psi_i = \varphi_{h(i)}$, for all $i$. $M$ operates in stages described as follows.

**Begin Stage $t$.** Input the datum $f(t)$ so that all the values of $f|_t$ are now known. For each $j \leqslant t$, simulate the computation of $\psi_j(x)$ for $t$ steps. For each $j \leqslant t$, let $a_j$, the *length of agreement*, be the largest number such that

(1) $a_j \leqslant t$ and
(2) $\psi_j(x)$ converges in at most $t$ steps for all $x \leqslant a_j$, and
(3) $\psi_j(x) = f(x)$ for all $x \leqslant a_j$.

$M$ outputs program $h(j)$ with belief level $1 - 1/2^{a_j}$, for each $j \leqslant t$.
**End Stage $t$.**

Since $\psi_0, \psi_1, \ldots$ is a Friedberg numbering, there is a unique $j$ such that $\psi_j = f$. Let $s$ be the sequence of output pairs produced by $M$ on input from the graph of $f$. Since $\psi_j = f$, program $h(j)$ is the output at all stages $t \geqslant j$. Furthermore, $E(s, h(j))$ is a monotonically nondecreasing sequence that approaches 1. Let $k$ be any number such that $k \neq j$. Then $\psi_k \neq \psi_j = f$. Choose the least $x$ such that $\psi_k(x) \neq f(x)$. Then, for stages $t \geqslant k$, the value $a_k < x$. Hence, $E(s, h(k))$ does not approach 1. Consequently, $M$ converges in the correct sense to $h(j)$ and so learns f. □

## 5. Learning formulas from elementary facts

We start with necessary logical concepts. A *model* will be a triple $\langle \Sigma, N, I \rangle$ where $\Sigma$ is a finite set of predicate symbols, called a *signature*, with designated arities, $N$ is the domain of the variables used in the predicates and $I$ is the interpretation of the predicates. Unless otherwise noted, the domain will be the natural numbers, **N**. For example, consider the model $M_0 = \langle \Sigma_0, \mathbf{N}, I_0 \rangle$ where $\Sigma_0$ contains three binary predicates, $P_1$, $P_2$ and $P_3$. The interpretation $I_0$ is given by three formulas: $P_1(x, y)$: $x \leqslant y$, $P_2(x, y)$: $y = 5x$ and $P_3(x, y)$: $y = x^2$. The *elementary facts* of a model are all of the instantiated predicate symbols of the model with associated truth values. The elementary facts of our example model $M_0$ include $P_1(2, 5) = T$, $P_1(6, 2) = F$, $P_1(4, 5) = T$, $P_2(2, 10) = T$, $P_3(3, 10) = F$. In some of the proofs that follow it will be more convenient to list these elementary facts as $P_1(2, 5)$, $\neg P_1(6, 2)$, $P_1(4, 5)$, $P_2(2, 10)$, $\neg P_3(3, 10)$.

Let $M = \langle \Sigma, \mathbf{N}, I \rangle$ be a model. By $E_M$ we denote the set of all elementary facts of the model $M$. By $D_M$ we denote the set of all enumerations of all elements of $E_M$.

By *elementary formulas* we understand formulas of type:

$$P(n_1, \ldots, n_k)$$

where $P$ is a $k$-ary predicate symbol from $\Sigma$ and $n_i$ $(i = 1, \ldots, k)$ is either variable or a notation for a domain element. For instance, if $P \in \Sigma$, then $P(x, y)$, $P(x, 7)$, $P(3, 7)$ are elementary formulas. By *formulas* we understand first-order predicate logic formulas (without functional symbols) constructed in the traditional way from the elementary formulas. For instance,

$$\forall x \exists y \big(P(x, y)\big),$$
$$\forall x \big(P(x, 7) \vee P(x, 9)\big)$$

are formulas. From now on we consider only closed formulas, i.e. formulas with no free variables. On any particular model such a formula is either true or false.

We are interested in the following problem: given an infinite sequence of all the elementary facts of the model $M$, to find out whether the formula is true or false in the model $M$.

### *5.1. Learning formulas in the limit*

We start considering formula learning process in the limit which is a counterpart to E.M. Gold's notion of learning in the limit.

**Definition 7.** *A limit formula learning machine* (abbreviated LFL) is a deterministic algorithmic device that takes as the input an enumeration of the elementary facts of some model and produces as the output a sequence of the first-order predicate logics formulas paired with values TRUE and FALSE.

An LFL can output the same formula several times, some instances with TRUE, some instances with FALSE. If $L$ is LFL and $D$ is its input data and $f$ is a predicate logic formula, then $S(L, D, f) = (a_1, a_2, \ldots)$ is the sequence of statements TRUE and FALSE associated with $f$.

**Definition 8.** We say that LFL $L$ learns a formula $f$ from the input data $D$ if and only if $S(L, D, f)$ is either a finite sequence with the last element TRUE or an infinite sequence with almost all elements TRUE. The set of formulas produced this way is denoted by $L(D)$.

**Definition 9.** An LFL $L$ is correct iff for all models $M$ and all $D \in D_M$, all the formulas in $L(D)$ are true for the model $M$.

**Definition 10.** We say that LFL $L$ learns formula $f$ iff $L$ learns the formula $f$ from every $D \in D_M$ of every model $M$ such that $f$ is true in $M$.

**Definition 11.** We say that LFL $L$ decides formula $f$ iff $L$ learns both $f$ and $\neg f$.

Of course, we study learning and deciding of formulas only by correct LFL $L$.

**Theorem 8.** *Let $A$ be an arbitrary quantifier-free formula with predicate symbols $P_1, \ldots, P_n$ from signature $\Sigma$, and $x_1, \ldots, x_u, y_1, \ldots, y_v$ be variables of the involved predicates. Then the formula*

$$\exists x_1 \ldots x_u \forall y_1 \ldots \forall y_v \big(A(P_1, \ldots, P_n, x_1, \ldots, x_u, y_1, \ldots, y_v)\big)$$

*is learnable by a correct LFL.*

**Proof.** Let $\alpha_0, \alpha_1, \alpha_2, \ldots$ be an enumeration of all possible $u$-tuples of nonnegative integers and $\beta_0, \beta_1, \beta_2, \ldots$ be an enumeration of all possible $v$-tuples of nonnegative integers. LFL systematically takes $u$-tuples $\alpha_0, \alpha_1, \alpha_2, \ldots$ and for the current $\alpha_i$ outputs FALSE immediately followed by TRUE, goes on in systematic consideration of all $v$-tuples until the formula $A$ turns out to be false. Then LFL goes to another $u$-tuple $\alpha_{i+1}$. If the formula is true, only a finite number of outputs related to this formula is output, and the last one is TRUE.Hence the LFL learns the formula If the formula is false, the sequence of outputs is infinite. Hence the LFL is correct. □

**Theorem 9.** *There is no correct LFL which learns formula* "$\forall x \exists y (P(x, y))$."

**Proof.** Assume from the contrary that this formula (called $f$ for the sequel) is learnable by a correct LFL $L$. We construct a predicate $P_L(x, y)$ such that the formula $\forall x \exists y (P_L(x, y))$ is true but it is not learnable by $L$.

Consider an effective numbering of all pairs of natural numbers $p(i), i = 0, 1, \ldots$. We denote the first component $x$ of the pair $p(i) = (x, y)$ by $x(i)$. We say that the set of pairs $S = \{p(s_1), \ldots, p(s_e)\}$ $x$-covers the set of pairs $R = \{p(r_1), \ldots, p(r_m)\}$ if $\{x(s_1), \ldots, x(s_e)\} \supseteq \{x(r_1), \ldots, x(r_m)\}$. We define the predicate $P_L$ stepwise.

**Step 1.** Make $P_L(p(0)), P_L(p(1)), \ldots, P_L(p(n_1))$ false until $L$ outputs the first FALSE for $\forall x \exists y(P(x, y))$.

**Step 2i.** Make $P_L(p(n_{2i-1}+1)), \ldots, P_L(p(n_{2i-1}+2)), \ldots, P_L(p(n_{2i}))$ true until

(1) $\{p(n_{2i-1}+1), p(n_{2i-1}+2), \ldots, p(n_{2i})\}$ $x$-covers $\{p(0), p(1), \ldots, p(n_{2i-1})\}$, and
(2) $L$ outputs TRUE for $\forall x \exists y(P(x, y))$.

**Step 2i + 1.** Make $P_L(p(n_{2i}+1)), P_L(p(n_{2i}+2)), \ldots, P_L(p(n_{2i+1}))$ false until $L$ outputs FALSE for $\forall x \exists y(P(x, y))$.

Since $L$ learns $f$, the above mentioned mind changes at the moments $n_1, n_2, n_3, \ldots$ exist, and the definition of $P_L$ is correct. It follows from the definition of $P_L$ that the formula $\forall x \exists y(P_L(x, y))$ is true. However, $L$ makes infinitely many mind changes. □

**Corollary 10.** *The formula "$\exists x \forall y(P(x, y))$" is learnable, but not decidable by an LFL.*

**Definition 12.** A correct LFL $L$ is called best iff $L'(D) \subseteq L(D)$ holds for arbitrary correct LFL $L'$, for arbitrary model $M$ and arbitrary enumeration of elementary facts $D \in D_M$.

**Theorem 11.** *There is no best LFL.*

**Proof.** Assume from the contrary that a best LFL exists. Denote this LFL by $L$. Consider the formula "$\forall x \exists y(P(x, y))$." Let $U$ denote the set of all the interpretations of the predicate $P(x, y)$ such that $\forall x \exists y(P(x, y))$ is true and $L$ learns $\forall x \exists y(P(x, y))$ at these interpretations. It follows from Theorem 9 that $U$ does not include all the interpretations of $P(x, y)$ for which $\forall x \exists y(P(x, y))$ is true.

Construct a predicate $P_L(x, y)$ by the method described in the proof of Theorem 9. Evidently $P_L(x, y)$ is an interpretation not in $U$ but $\forall x \exists y(P_L(x, y))$ is true. It is easy to see from the consideration of the construction of $P_L(x, y)$ that there is an algorithm (depending on $L$) to compute the values of $P_L(x, y)$. This implies the existence of a computable function $G_L(x)$ such that $\forall x P_L(x, G_L(x))$ is true. By using the function $G_L(x)$, it is easy to construct a correct LFL $L'$ learning $\forall x \exists y(P(x, y))$ at the interpretation $P_L(x, y)$. Hence, there is an interpretation for which $L$ does not learn a formula, but another LFL $L'$ does the job. Contradiction with the assumption $L$ being a best LFL. □

**Theorem 12.** *If $f$ is a first-order predicate logics formula involving only monadic predicates, then $f$ is decidable by correct LFL.*

**Proof.** Without loss of generality, we assume that the target formula $f$ is in prenex form. We proceed by induction. For formulas without quantifiers the theorem is obvious. Assume that the theorem is true for $n-1$ quantifiers and the target formula $f$ contains $n$ quantifiers.

Let $Qx$ be the outermost quantifier of the formula $f$. Suppose that $P_1, \ldots, P_m$ is a complete list of all monadic predicates in $f$ which contain the variable $x$. To simplify the proof assume that $m = 2$. The generalization to larger values of $m$ is obvious. We define 4 formulas $f_1$, $f_2$, $f_3$, $f_4$ (in general $2^m$ formulas) derived from $f$ substituting, respectively, $P_1(x) = T$, $P_2(x) = T$ for $f_1$, $P_1(x) = T$, $P_2(x) = F$ for $f_2$, $P_1(x) = F$, $P_2(x) = T$ for $f_3$, $P_1(x) = F$, $P_2(x) = F$ for $f_4$. It is important that $f_1$, $f_2$, $f_3$, $f_4$ have at most $n-1$ quantifiers. The target formula is equivalent to

$$Qx\big(\big(P_1(x) \wedge P_2(x) \wedge f_1\big) \vee \big(P_1(x) \wedge \neg P_2(x) \wedge f_2\big) \vee \big(\neg P_1(x) \wedge P_2(x) \wedge f_3\big) \vee \big(\neg P_1(x) \wedge \neg P_2(x) \wedge f_4\big)\big).$$

By the induction assumption there are LFLs $L_1, L_2, L_3, L_4$ deciding $f_1$, $f_2$, $f_3$, $f_4$ respectively. If $Qx$ is the existential quantifier $\exists x$, the machine $L$ finds out in the limit which of the four formulas $g_1$: $\exists x(P_1(x) \wedge P_2(x))$, $g_2$: $\exists x(P_1(x) \wedge \neg P_2(x))$, $g_3$: $\exists x(\neg P_1(x) \wedge P_2(x))$, $g_4$: $\exists x(\neg P_1(x) \wedge \neg P_2(x))$ are true in the model $M$. These formulas are decidable by LFL $K_1$, $K_2$, $K_3$ and $K_4$. In parallel $L$ outputs the disjunction of the outputs of all the machines $L_1$, $L_2$, $L_3$ and $L_4$ with the corresponding indices.

If $Qx$ is the universal quantifier $\forall x$, the machine $L$ finds out in the limit which of the four machines $L_1$, $L_2$, $L_3$, $L_4$ output TRUE (this subset stabilizes in the limit), reacting to every mind change by outputting the sequence TRUE, FALSE, and outputs the disjunction of the outputs of all the machines $K_1$, $K_2$, $K_3$ and $K_4$ with the corresponding indices. □

### 5.2. Learning formulas with belief levels

Now we consider another learning process which is essentially infinite. The machine attaches to each output a belief level. For some formula the belief level may grow while for some other formula the belief level stabilizes at some intermediate level.

**Definition 13.** *Belief formula learning machine* (abbreviated BFL) is a deterministic algorithmic device that takes as the input an enumeration of the elementary facts of some model and produces as the output an infinite sequence of first-order predicate logic formulas paired with belief levels.

A BFL will produce a sequence of outputs, where the same formula may appear over and over again, with different belief levels. If $B$ is BFL and $D$ is its input data and $f$ is a predicate logic formula, then $Z(B, D, f) = (b_1, b_2, \ldots)$ is the sequence of belief levels associated with $f$ that $B$ produces when given $D$ as input, in the order of appearance in the output stream.

**Definition 14.** We say that a BFL $B$ learns a formula $f$ from the input data $D$ if and only if $Z(B, D, f)$ monotonically converges to 1. The set of formulas produced this way is denoted by $B(D)$.

**Definition 15.** A BFL $B$ is correct iff for all models $M$ and all $D \in D_M$, all the formulas in $B(D)$ are true for the model $M$.

**Definition 16.** We say that a BFL $L$ learns formula $f$ iff $B$ learns formula $f$ from every $D \in D_M$ of every model $M$ such that $f$ is true in $M$.

For instance, the formula $\forall x(P(x))$ is learnable by a correct BFL. Indeed, let $\omega(n) = 1/2^{n+1}$. If BFL finds out that $P(0) \wedge P(1) \wedge \cdots \wedge P(n)$ is true, BFL outputs the formula $\forall x(P(x))$ with belief level $\Sigma_{i=0}^{n}\omega(i)$. Hence, the BFL learns $\forall x(P(x))$ iff it is true for the considered model.

**Definition 17.** We say that a BFL $B$ decides formula $f$ iff $B$ learns both $f$ and $\neg f$.

**Theorem 13.** *Let $A$ be an arbitrary quantifier-free formula with predicate symbols $P_1, \ldots, P_n$ from signature $\Sigma$, and $x_1, \ldots, x_u, y_1, \ldots, y_v$ be variables of the involved predicates. Then the formula*

$$\forall x_1, \ldots, \forall x_u, \exists y_1, \ldots, \exists y_v A(P_1, \ldots, P_k, x_1, \ldots, x_u, y_1, \ldots, y_v)$$

*is learnable by a correct BFL.*

**Proof.** Let $\alpha_0, \alpha_1, \alpha_2, \ldots$ be an enumeration of all possible $u$-tuples of nonnegative integers and $\beta_0, \beta_1, \beta_2, \ldots$ be an enumeration of all possible $v$-tuples of nonnegative integers. Let $\omega(n) = 1 - 1/2^{n+1}$. If the BFL finds out that for $u$-tuples $\alpha_0, \alpha_1, \alpha_2, \ldots$ exists $v$-tuples $\beta_{i_0}, \beta_{i_1}, \ldots, \beta_{i_n}$ such that $A(P_1, \ldots, P_k, \alpha_0, \beta_{i_0}), A(P_1, \ldots, P_k, \alpha_1, \beta_{i_1}), \ldots, A(P_1, \ldots, P_k, \alpha_n, \beta_{i_n})$ are true, then the BFL outputs the formula $\forall x_1, \ldots, \forall x_u, \exists y_1, \ldots, \exists y_v A(P_1, \ldots, P_k, x_1, \ldots, x_u, y_1, \ldots, y_v)$ with belief level $\Sigma_{i=0}^{n}\omega(i)$. Hence, the BFL learns this formula iff the formula is true for the considered model. □

**Theorem 14.** *There is no correct BFL which learns formula* "$\exists x \forall y(P(x, y))$."

**Proof.** Assume by way of contradiction that a BFL $L$ learns $f = \exists x \forall y(P(x, y))$. We construct, in effective stages of finite extension, an interpretation for the predicate $P(x, y)$ such that $L$ outputs $f$ with belief levels converging to 1, but the formula $f$ is not true under this interpretation of the predicate. Each stage $s$ will employ an auxiliary predicate $P_s$ that is defined prior this stage, and $\exists x \forall y(P_s(x, y))$ is true. By way of initialization, $P_0$ is the predicate over two arguments that is always true. We define an increasing sequence of integers $m_s$ to mark boundaries of areas where $P$ is equal to $P_s$. The first integer $m_0$ is defined to be 0. Execute the following stages for $s = 0, s = 1, \ldots$.

**Begin stage $s$.** Use values of $P_s$ as the elementary facts for $L$ until $L$ produces an output $f/e$ for $e \geqslant 1 - 1/2^{s+1}$. Since $\exists x \forall y (P_s(x, y))$ is true, such an output will eventually be produced. Let $D_s$ be the set of pairs $(x, y)$ such that the value $P_s(x, y)$ was read from the input by $L$ to produce $f/e$. Of course, $D_s$ is a finite set. Let $m_{s+1}$ be the largest value $x$ in pairs $(x, y) \in D_s$. Define $P_{s+1}$ as follows

$$P_{s+1}(x, y) = \begin{cases} T, & \text{if } (x, y) \in D_s; \\ T, & \text{if } x > m_{s+1}; \\ F, & \text{otherwise.} \end{cases}$$

**End stages.**

Define $P(x, y) = P_s(x, y)$ for $m_{s-1} < x \leqslant m_s$, $s = 1, 2, \ldots$ Notice that $P$ coincides with $P_s$ on all the values used by $L$ to produce $f/e$ with some $e \geqslant 1 - 1/2^s$. Hence, when the values of $P$ are used as the elementary facts for $L$, $L$ will produce belief levels for $f$ converging to 1. In other words, $L$ learns $f$. However, for an arbitrary $x$, and for all sufficiently large $y$'s, $P(x, y) = F$. Hence $f$ is false for this interpretation of $P$, a contradiction. □

**Theorem 15.** *If a formula is decidable by LFL, then it is decidable by BFL as well.*

**Proof.** Since the formula $f$ is decidable by LFL $L$, the outputs concerning $f$ stabilize on TRUE or FALSE. Hence the correct answer is the current output at infinitely many points, while the incorrect answer is the current output at only finitely often. Hence the BFL can raise expectation level for the current output of the LFL. □

**Theorem 16.** *If $f$ is a first-order predicate logic formula involving only monadic predicates, then $f$ is decidable by BFL.*

**Proof.** Immediately from Theorems 12 and 15. □

**Definition 18.** A correct BFL $B$ is called *best* iff $B'(D) \subseteq B(D)$ holds for an arbitrary correct BFL $B'$, for an arbitrary model $M$ and an arbitrary enumeration of elementary facts $D \in D_M$.

**Theorem 17.** *There is no best BFL.*

**Proof.** The proof starts by the construction of a sequence of BFLs $B_0, B_1, B_2, \ldots$. The idea is to make each of the BFLs behave correctly and in particular the following property holds for any model $M$ and any enumeration of elementary facts $D \in D_M$, $Z(B_i, D, \exists x \forall y P(x, y))$ monotonically approaches 1 iff $\forall y (P(i, y))$ is true in $M$.

The machine $B_i$ is really quite simple. Whenever it finds elementary facts $P(i, 0) = T$, $P(i, 1) = T, \ldots,$ $P(i, n) = T$, the machine outputs the formula $\exists x \forall y P(x, y)$ with belief level $\Sigma_{i=0}^{n} \omega(i)$ where $\omega(i) = 1/2^{i+1}$.

Choose any model $M$ for which $\exists x \forall y P(x, y)$ is true. Then, in the model $M$, $\forall y P(i, y)$ is true for some $i$. Hence BFL $B_i$ learns this formula for model $M$.

Suppose by way of contradiction that $B$ is the best BFL. Then it covers BFL's $B_0, B_1, B_2, \ldots$. Hence $B$ learns the formula $\exists x \forall y P(x, y)$ for every model $M$. A contradiction to Theorem 14. □

## 6. A logic of discovery

This section is the central one in our paper. We will study the axiomatization of learning formulas with belief levels. We start with introducing a new notion

$$A/e$$

where $A$ is a formula and $e$ is a rational number with $0 \leqslant e \leqslant 1$. We call the number $e$ the *belief level* of the formula $A$. The intuitive meaning of the belief level $e$ is that, based on the prior steps of the proof, we have belief $e$ that the formula $A$ is true.

To formulate the new axioms we need one more notion, namely, that of weighting function

**Definition 19.** A *weighting function* is a recursive function $\omega$ such that

$$\sum_{n=1}^{\infty} \omega(n) = 1.$$

For example, let $\omega(n) = 1/2^n$. The idea of the weighting function is that some examples are more relevant than the others. The weighting function will be used to determine the belief level of a universally quantified sentence, based on known examples.

Our proof system starts with a standard first-order theory (see, e.g., [17]). Proofs are built in the standard fashion. We allowed to write $A$ as a line of a proof if $A$ can be deduced by the usual axioms from prior lines in the proof, or we have been given that $A$ is true as an input to the discovery process.

We start by introducing the *t-axiom*, or *truth* axiom:

$$\frac{A}{A/1}.$$

This axiom merely says that if $A$ is provable within the standard first-order theory, then $A$ is provable with belief 1 in our system.

We continue by adding to the standard set of axioms for the first-order theory the following *e-axiom* or *belief* axiom for a formula $A$ with one free variable:

$$\frac{A(n_1)/e_1, \ldots, A(n_k)/e_k}{\forall x A(x)/\sum_{i=1}^{k} \omega(n_i) \cdot e_i}.$$

Intuitively, if we have reasoned that $A$ is true on some examples (the $n_i$'s), then we have some belief that $(\forall x)A(x)$ is true and this belief is proportional to the number of examples and the confidence that we have in those examples. In our examples below we use the t-axiom so that all of the $e_i$'s associated with the $n_i$'s are 1. If in some lines of a proof we have $A(2)$ and then later $A(4)$, we would be able to use the e-axiom to obtain $(\forall x)A(x)/e$ where $e = \omega(2) + \omega(4)$.

Sometimes, in our proof system, we may derive a formula $A$ with some belief $e$ and then, from $A$ derive $B$, using only traditional logic. We would like the belief of $A$ to convey to $B$, but some care must be taken in doing so. To describe the conveyance, we first must define a notion of syntactic complexity for formulas.

Unless explicitly stated otherwise, below by formulas we understand formulas in prenex form.

**Definition 20.** Formula $A$ is no more complex than $B$ (written $A \preccurlyeq B$) iff

(1) $A$ has no more quantifiers than $B$, and
(2) $A$ has no more constants than $B$, i.e. there is no constant which is used in $A$, but not used in $B$.

For example, consider formulas

$$A_1\colon \quad \forall x \forall y \exists z \exists u \big(R(17, x, z) \wedge \big(\neg P(u, 5) \vee S(y)\big)\big),$$
$$A_2\colon \quad \exists x \exists v \exists w \exists z \big(\big(G(5, w) \vee H(v, 17, x)\big) \wedge \neg K(x, 17)\big),$$
$$A_3\colon \quad \forall g H(g, g, 17).$$

It is easy to see that $A_1 \preccurlyeq A_2$, $A_2 \preccurlyeq A_1$, $A_3 \preccurlyeq A_1$, $A_3 \preccurlyeq A_2$, but it is not the case that $A_1 \preccurlyeq A_3$, $A_2 \preccurlyeq A_3$.

Please notice that our definition of $\preccurlyeq$ ignores the names of the bounded variables.

We now introduce the *c-axiom*, or *conveyance* axiom. Let $A \Rightarrow B$ denote the situation where $B$ follows from $A$ in the traditional first-order logic.

$$\frac{A/e,\ A \Rightarrow B \text{ and } A \preccurlyeq B}{B/e}.$$

**Definition 21.** We say that $A/e$ is e-provable in the model $M$ iff starting from elementary facts of model $M$ and using traditional first-order theory argumented with the c-axiom, the e-axiom and the t-axiom we can obtain $A/e$.

We proceed to give as an example, a portion of an e-proof. We assume that the weight function is $\omega(n) = 1/2^{n+1}$.

| | |
|---|---|
| $A(1)$ | elementary fact (input), |
| $\exists x A(x)$ | 1, traditional logic, |
| $A(1)/1$ | 1, t-axiom, |
| $\forall x A(x)/.25$ | 3, e-axiom, |
| $A(3)$ | elementary fact (input), |
| $A(3)/1$ | 5, t-axiom, |
| $\forall x A(x)/.3125$ | 3, 6, e-axiom, |
| $\forall x A(x) \quad \Rightarrow \quad \forall x \forall y\big(A(x) \vee B(y)\big)$ | traditional logic, |
| $\forall x \forall y\big(A(x) \vee B(y)\big)/3125$ | 7, 8, c-axiom. |

**Definition 22.** We say that $A$ is e-provable in the model $M$ iff for every $\epsilon > 0$ there is an $e \geqslant 1 - \epsilon$ such that $A/e$ is e-provable in the model $M$.

**Definition 23.** We say that the axiom system $\Omega$ is sound for formula learning with belief levels iff every formula which is e-provable in the model $M$ by $\Omega$ is true in the model $M$.

**Theorem 18.** *The traditional first-order logic plus the e-axiom, the c-axiom and the t-axiom is sound.*

**Proof.** Our formulas are in prenex form. The quantifierless body of the formula is a Boolean propositional formula made of terms being predicate symbols from the finite signature $\Sigma$ with constants and variable symbols.

Consider, for instance, two formulas

$$A = \forall x \exists y\big(P(x, 7, y) \;\Rightarrow\; Q(y, 5)\big),$$
$$B = \forall x \exists y\big(Q(y, 5) \vee \neg P(x, 7, y)\big).$$

They have the same quantifier prefix (with equal names for the bounded variables) and the quantifierless bodies are equivalent Boolean formulas of the same variables $P(x, 7, y)$ and $Q(y, 5)$. It is easy to conjecture that $A$ is e-provable in a model $M$ if and only if $B$ is e-provable in $M$. To prove such a conjecture we consider the notion of *reversible equivalence*.

**Definition 24.** We say that formulas $A$ and $B$ are *reversibly equivalent* ($A \equiv_{\text{rev}} B$) if they have the same quantifier prefix (with equal names for the bound variables) and the quantifier less bodies are equivalent Boolean formulas of the same variables.

**Definition 25.** We say that formulas $A$ and $B$ are *equivalent up to names* ($A \equiv_{\text{rename}} B$) if they differ only in the names of the bounded variables.

**Definition 26.** We say that formulas $A$ and $B$ are *generalized reversibly equivalent* ($A \equiv_{\text{grev}} B$) in the model $M$, if there is a formula $C$ such that $A \equiv_{\text{rev}} C$, and $C \equiv_{\text{rename}} B$.

**Lemma 2.** *If in $M$ one can e-prove $A/e$, and $A \equiv_{\text{grev}} B$, then one can also e-prove $B/e$ in $M$.*

**Proof.** $A \equiv_{\text{grev}} B$ implies $A \Rightarrow B$ in traditional logic, and $A \preccurlyeq B$. Hence, by the c-axiom, $B/e$ is e-provable in $M$ if $A/e$ is e-provable in $M$. □

Let the formula $A$ be $\forall x F(x)$, and let $\Pi$ be an $e$-proof of $A/e$. The last axiom used in proof $\Pi$ can be either t-axiom or c-axiom or e-axiom. If the last axiom is e-axiom, then we say that $A/e$ is obtained by *direct using* of e-axiom.

**Lemma 3.** *Let $A = \forall x F(x)$, and $B = \forall y G(y)$. If one can e-prove $A/e$ in $M$ by direct using of e-axiom and $A \equiv_{\text{grev}} B$, then one can e-prove $B/e$ in $M$ also by direct using of e-axiom.*

**Proof.** Let $a$ be arbitrary constant. Obviously, $F(a)$ and $G(a)$ are also generalized reversible equivalent. According to Lemma 2, $F(a)/e$ is e-provable in $M$ iff $G(a)/e$ is e-provable in $M$. From this follows our lemma. □

**Lemma 4.** *For an arbitrary formula $B$, there are only a finite number (up to generalized reversible equivalence) of formulas $A$ such that $A \preccurlyeq B$.*

**Proof.** Let $S$ be the set of the constants used in $B$. There is only a finite number of different Boolean formulas made of terms consisting of predicate symbols from the finite set $\Sigma$, the bounded variables used in $B$ and the constants from $S$. □

We complete the proof of the theorem by showing that for an arbitrary formula $B$ which is false in the model $M$, and for an arbitrary weighting system $\omega$, there is a real number $e_0 < 1$ such that e-proofs of $B/e$ in $M$ cannot exist for $e > e_0$. We proceed by induction over $k$, the number of quantifiers used in $B$.

The base case ($k = 0$) is when $B$ has no quantifiers, so e-proofs are also first-order proofs.

Suppose inductively that for any formula $A$ with at most $k$ quantifiers which is false in the model $M$, there is a real number $e_0 < 1$ such that there are no e-proofs of $A/e$ in $M$ for any $e > e_0$. Suppose further that $B$ is a formula with $k + 1$ quantifiers that is false in the model $M$, and we can e-prove $B/e$ in $M$ for arbitrarily high $e < 1$.

Consider all possible ways how to e-prove $B/e$. For $e < 1$ the t-axiom cannot be used. If the leading quantifier of $B$ is $\exists$, then the e-axiom cannot be used. Suppose the leading quantifier is $\forall$ and $B$ is the formula $\forall x F(x)$. Since $B$ is false by assumption, there must be an $a$ such that $F(a)$ is false. $F$ has exactly $k$ quantifiers, so by the induction hypothesis, $F(a)$ cannot be proved with arbitrarily high expectation values. This will prevent the e-axiom from producing $B/e$ with an arbitrarily large $e < 1$.

Let us pass over to c-axiom. Consider all the $A$ such that $A \preccurlyeq B$. Such $A$ are possible candidates for usage in c-axiom to obtain $B/e$. One of three cases applies:

**Case 1.** $A$ is true in $M$. Then, in $M$, there is no first-order proof of $A \Rightarrow B$. Hence, c-axiom is not applicable in this case.

**Case 2.** $A$ is false in $M$ and $A$ has at most $k$ quantifiers. Then, by the induction hypothesis, one cannot e-prove $A/e_1$ with arbitrarily high values of $e_1$. By Lemma 4 there are only a finite number of different (up to generalized reversible equivalence) formulas A. By Lemma 2, generalized reversible equivalent formulas are e-provable only up to the same belief level $e$. Hence there is a real number $e_0 < 1$ such that e-proofs of $A/e$ in $M$ cannot exist for $e > e_0$. Hence such $A$ cannot be used in c-axiom to obtain $B/e$ with arbitrarily high $e$.

**Case 3.** $A$ is false in $M$, and $A$ has exactly $k + 1$ quantifiers (this is the largest possible number of quantifiers for any formula $A \preccurlyeq B$). By Lemma 4 there are only a finite number of different (up to generalized reversible equivalence) such formulas A. Therefore, to prove by c-axiom $B/e$ for an arbitrary high value of $e$, at least one of these $A$ must be with similar feature, namely, it must be possible to prove $A/e$ for an arbitrarily high value of $e$. It is easy to see that c-axiom does not produce new belief levels, it only transfers these $e$ values from one formula to another formula. The only axiom which produces new belief levels is e-axiom. This axiom works only for formulas $A$ with leading quantifier $\forall$. As $A$ is false, then using reasoning similar to the previous case (when leading quantifier of $B$ was $\forall$), we conclude that there exists $\epsilon_A < 1$ such that it is not possible to prove with e-axiom $A/e$ for $e > \epsilon_A$. According to Lemma 3 the same refers to all formulas which are generalized reversible equivalent to $A$. As there are only a finite number of different (up to generalized reversible equivalence) such formulas $A$, it means that (by Lemma 4) there exists $\epsilon_0 < 1$ such that for all $A$ there are $e_A \leqslant \epsilon_0$. From this follows that we cannot prove with c-axiom $B/e$ for $e > \epsilon_0$. □

**Theorem 19.** *Suppose $\omega_1$ and $\omega_2$ are two different weighting functions. Then, for any formula $A$, $A$ is e-provable with respect to $\omega_1$ iff $A$ is e-provable with respect to $\omega_2$.*

**Proof.** If the formula $A$ is of form $\forall x\, B(x)$ and its proof, in addition to classical axioms of logic and t-axiom, uses only the e-axiom, then it is obvious that if $A$ is e-provable with respect to one weight function, then it is e-provable with respect to any other weight function.

The case when the proof of formula $A/e$ uses also the c-axiom is more complicated. For the sake of simplicity let as assume that the c-axiom is used only in the final stage of proof of $A/e$ (in the general case, if $A$ is of form $\forall x\, B(x)$, the c-axiom may have been used also in the proof of $B(a)/e$). According to the c-axiom, we can transfer the belief level $e$ along the sequence of implications $A_1 \Rightarrow A_2 \Rightarrow \cdots \Rightarrow A_n$, if $A_1 \preccurlyeq A_2 \preccurlyeq \cdots \preccurlyeq A_n$, and in the result we can obtain $A/e$ from $A_1/e$. If we consider this sequence of implications from the very beginning, then $A_1/e$, if $e < 1$, can be obtained only by direct using of e-axiom. Consequently, $A_1$ must be of the form $\forall x\, B_1(x)$. At the same time, according to Lemma 4, there are only a finite number (up to generalized reversible equivalence) of formulas $A_i$ such that $A_i \preccurlyeq A$. Since $A$ is e-provable, then it follows that there must be at least one (up to generalized reversible equivalence) $A_1 = \forall x\, B_1(x)$ such that $A_1 \preccurlyeq A$, $A_1 \Rightarrow A$, and for this $A_1$ by direct using of e-axiom a sequence of assertions $A_1/e_1, A_1/e_2, \ldots, A_1/e_n, \ldots$ can be proven, where $e_n \to 1$ when $n \to \infty$. Hence, by c-axiom it follows that $A/e_1, A/e_2, \ldots, A/e_n, \ldots$. However, if for $A_1 = \forall x\, B_1(x)$ by direct using of e-axiom a sequence of assertions $A_1/e_1, A_1/e_2, \ldots, A_1/e_n, \ldots$ where $e_n \to 1$ when $n \to \infty$ can be proved with respect to one weight function, then similar assertions, obviously, can be proved for any other weight function. Consequently, if we can e-prove the formula $A$ with respect to one weight function, then we can e-prove $A$ with respect to any other weighting function. □

## 7. Justification of the new axioms

The three new axioms we introduced above, with the exception of (technically needed) Definition 20, are natural enough, but are they presented here in the most concise form? The t-axiom is the least controversial. It merely takes truths from the traditional domain and maps them to statements with the highest possible expectation of truth in our logic of discovery. The e-axiom provides a way of calculating a belief level, modulo a weighting system, based on prior discoveries. Notice that the use of the e-axiom is the only way to generate a belief level less than 1. The c-axiom is a form of Modus Ponens. Contrary to traditional Modus Ponens, it contains additional condition $A \preccurlyeq B$. Perhaps a simpler version (without this additional condition) might suffice? To show that nothing obviously simpler will suffice, we consider the ramifications of some tempting simpler schemes. For the sake of discussion, consider the following version of the c-axiom with one condition ($A \preccurlyeq B$) removed:

$$\frac{A/e,\ A \Rightarrow B}{B/e}.$$

We denote this version as $c_1$-*axiom*.

**Theorem 20.** *Traditional first-order logic plus t-axiom, the e-axiom and the* $c_1$*-axiom is NOT sound.*

**Proof.** We prove that there is a model $M$ and a formula $A$ such that $A$ is false in the model $M$ but nonetheless $A$ is e-provable in $M$ via this system of axioms.

Indeed, we consider a model $M$ with one unary predicate $P(x)$ such that $P(1)$ is true and $P(2)$ is false. We consider the formula $\forall x\, P(x)$ which is false in $M$. Let $\omega(n)$ be the weighting function. Our e-axiom provides us $\forall x\, P(x)/\omega(1)$.

We construct a formula $Q_k$ being

$$\forall x_1 \forall x_2 \cdots \forall x_k \big(P(x_1) \vee P(x_2) \vee \cdots \vee P(x_k)\big).$$

A somewhat similar formula being

$$\forall x_1 \forall x_2 \cdots \forall x_k \big(P(x_1) \vee P(x_2) \vee \cdots \vee P(x_k)\big) \quad \Rightarrow \quad \forall x\, P(x)$$

is a tautology in the standard predicate logic. Hence, by t-axiom, it can be e-proved with belief level 1. Using e-axiom $k$ times, we obtain

$$Q_k/1 - \big(1 - \omega(1)\big)^k.$$

Applying $c_1$-axiom, we get

$$\forall x\, P(x)/1 - \big(1 - \omega(1)\big)^k.$$

If $k \to \infty$, then the belief level converges to 1. Hence the false formula $\forall x P(x)$ is e-provable with belief level converging to 1. □

As a consequence of Theorem 20, a straightforward generalization of Modus Ponens fails soundness. We know from Theorem 18 that imposing some relative complexity constraint on pairs of formulas result in a sound logic. Will perhaps a simpler version of Definition 20 suffice? We consider dropping the second condition of Definition 20 that the more complex formula have at least as many distinct constants. We denote the axiom that results from using the c-axiom, except referring to this relaxed version of relative formula complexity, as $c_2$-axiom.

**Theorem 21.** *Traditional first-order logic plus t-axiom, the e-axiom and the $c_2$-axiom is NOT sound.*

**Proof.** We consider a model $M$ where the binary predicate $P(x, y)$ means "$y \leqslant x$." In this model the formula $\exists x \forall y P(x, y)$ is false. We consider the formula $\forall y P(x, y)$ with one free variable $x$. Let $k$ be an arbitrary natural number. Using e-axiom, we can get the belief level

$$e_k = \omega(1) + \omega(2) + \cdots + \omega(k)$$

for the formula $\forall y P(k, y)$. On the other hand, in the classical logic the formula

$$\forall y P(k, y) \quad \Rightarrow \quad \exists x \forall y P(x, y)$$

is a tautology. Using $c_2$-axiom with $A = \forall y P(k, y)$ and $B = \exists x \forall y P(x, y)$ we get the belief level $e_k = \omega(1) + \omega(2) + \cdots + \omega(k)$ for the false formula $\exists x \forall y P(x, y)$. Since $k \to \infty$ implies $e_k \to 1$, we get belief level converging to 1 for a false formula $\exists x \forall y P(x, y)$. □

## 8. Power of the axiom system

This section shows that our axiom system is powerful enough to e-prove the same classes of formulas which are learnable by BFL according Theorems 12 and 13.

**Definition 27.** We say that formula $A$ is e-provable iff $A$ is e-provable in every model $M$ such that $A$ is true in $M$.

**Theorem 22.** *Let $A$ be an arbitrary Boolean formula with predicate symbols $P_1, \ldots, P_n$, and $x_1, \ldots, x_u, y_1, \ldots, y_v$ be variables of the involved predicates. Then the formula*

$$\forall x_1 \cdots \forall x_u \exists y_1 \cdots \exists y_v A(x_1, \ldots, y_v)$$

*is e-provable.*

**Proof.** We prove here only a special case of this theorem when the formula is $\forall x \exists y P(x, y)$ and there is a sequence $\{a_1, a_2, a_3, \ldots\}$ of natural numbers such that $P(1, a_1)$ is true, $P(2, a_2)$ is true, etc. The general case is similar and differs only in more heavy technical notation. The following sequence of proofs is performed via our axiom system.

| | |
|---|---|
| $P(1, a_1)$ | elementary fact, |
| $\exists y P(1, y)$ | traditional logic, |
| $\exists y P(1, y)/1$ | t-axiom, |
| $\forall x \exists y P(x, y)/\omega(1).1$ | e-axiom, |
| $P(2, a_2)$ | elementary fact, |
| $\exists y P(2, y)$ | traditional logic, |
| $\exists y P(2, y)/1$ | t-axiom, |
| $\forall x \exists y P(x, y)/\omega(1).1 + \omega(2).1$ | e-axiom, |
| $P(3, a_3)$ | elementary fact, |
| $\exists y P(3, y)$ | traditional logic, |

$$\begin{array}{ll} \exists y P(3,y)/1 & \text{t-axiom,} \\ \forall x \exists y P(x,y)/\omega(1)+\omega(2)+\omega(3) & \text{e-axiom,} \\ \vdots & \vdots \quad \square \end{array}$$

**Theorem 23.** *If $A$ is a first-order predicate logic formula involving only monadic predicates, then $A$ is e-provable.*

**Proof.** Let us remember that we consider only formulas in prenex form. We prove our theorem by induction. For quantifierless formulas the theorem is obvious. Assume that the theorem is true for $k$ quantifiers and the target formula $F$ contains $k+1$ quantifiers.

Let $Qx$ be the outermost quantifier of the formula $F$. Suppose that $P_1, \ldots, P_m$ is a complete list of all monadic predicates in $F$, those which contain the variable $x$. To simplify the proof, assume that $m = 2$. The generalization to larger values of $m$ is obvious. We define 4 formulas $f_1$, $f_2$, $f_3$, $f_4$ (in general $2^m$ formulas) derived from $F$ substituting, respectively, $P_1(x) = T$, $P_2(x) = T$ for $f_1$, $P_1(x) = T$, $P_2(x) = F$ for $f_2$, $P_1(x) = F$, $P_2(x) = T$ for $f_3$, $P_1(x) = F$, $P_2(x) = F$ for $f_4$. It is important that $f_1$, $f_2$, $f_3$, $f_4$ have at most $k$ quantifiers. The target formula $F$ is equivalent to $Qx((P_1(x) \,\&\, P_2(x) \,\&\, f_1) \vee (P_1(x) \,\&\, \neg P_2(x) \,\&\, f_2) \vee (\neg P_1(x) \,\&\, P_2(x) \,\&\, f_3) \vee (\neg P_1(x) \,\&\, \neg P_2(x) \,\&\, f_4))$.

We consider the cases $Q = \exists$ and $Q = \forall$ separately.

**Case 1.** Assume that $F$ is $\exists x G(x)$, and it is true in the model $M$ for $x = 7$. Then for $x = 7$ only one part out of four

$$\begin{aligned} &\big(P_1(x) \,\&\, P_2(x) \,\&\, f_1\big) \\ &\quad \vee \big(P_1(x) \,\&\, \neg P_2(x) \,\&\, f_2\big) \\ &\quad \vee \big(\neg P_1(x) \,\&\, P_2(x) \,\&\, f_3\big) \\ &\quad \vee \big(\neg P_1(x) \,\&\, \neg P_2(x) \,\&\, f_4\big) \end{aligned}$$

can be true in $M$. For instance, let $P_1(7) \,\&\, \neg P_2(7) \,\&\, f_2$ is true in $M$. This implies that $f_2$ is true in the model $M$ for all the values of $x$ (because $f_2$ does not contain $x$). By induction assumption, $f_2/e$ is e-provable for arbitrarily high $e < 1$. Consider c-axiom with $A = f_2$ and $B = \exists x G(x)$. Since $f_2 \Rightarrow \exists x G(x)$ is true in the model $M$, and our *classical* proofs of the validity of formulas in models use all the power of Kleene axiom system, it is possible to prove

$$f_2 \quad \Rightarrow \quad \exists x F(x).$$

Beside that,

$$f_2 \preccurlyeq \forall x G(x).$$

Hence, by c-axiom, $\exists x G(x)/e$ for an arbitrarily high $e < 1$.

**Case 2.** Assume that $F$ is $\forall x G(x)$, and it is true in the model $M$. It means that for every $x$ separately (e.g., for $x = 7$)

$$\begin{aligned} G(7) \equiv \big[&\big(P_1(7) \,\&\, P_2(7) \,\&\, f_1\big) \\ &\vee \big(P_1(7) \,\&\, \neg P_2(7) \,\&\, f_2\big) \\ &\vee \big(\neg P_1(7) \,\&\, P_2(7) \,\&\, f_3\big) \\ &\vee \big(\neg P_1(7) \,\&\, \neg P_2(7) \,\&\, f_4\big)\big] \end{aligned}$$

can be true in $M$. From this follows that at least one of these 4 parts must be true. If, for instance, the part

$$\big(P_1(7) \,\&\, \neg P_2(7) \,\&\, f_2\big)$$

is true, then $f_2$ (not containing $x$) is true, and

$$f_2 \quad \Rightarrow \quad G(7).$$

By inductive assumption, $f_2/e$ is e-provable in $M$ for arbitrary high $e < 1$. In the same time

$$f_2 \preccurlyeq G(7).$$

Hence, by c-axiom we obtain $G(7)/e$ for arbitrary high $e < 1$. Similar property is valid for every natural number (not only for 7). Now, using e-axiom, we obtain

$$\frac{G(1)/e_1, \ldots, G(k)/e_k}{(\forall x)G(x)/e}$$

where $e = \sum_{i=1}^{k} \omega(i) \cdot e_i \to 1$ when $k \to \infty$ and $e_i \to 1$. □

## 9. Is this axiom system complete?

There are at least two natural definitions of completeness of axiom systems for formula learning with belief levels.

**Definition 28.** We say that axiom system $\Omega$ is *absolutely complete* for formula learning with belief levels iff for every correct BFL $L$, every model $M$ and every enumeration of elementary facts $D \in D_M$ there holds: if $A \in L(D)$, then $A$ is e-provable by $\Omega$ in the model $M$.

It is easy to see that there do not exist absolutely complete axiom systems for formula proving with belief levels. Indeed, it is easy to observe that an arbitrary axiom system for formula learning with belief levels can be simulated by a BFL machine such that the machine learns all the formulas e-provable via this axiom system. An absolutely complete axiom system would generate a best correct BFL machine which is impossible because of Theorem 17.

**Definition 29.** We say that axiom system $\Omega$ is *practically complete* for formula learning with belief levels if every formula which is e-learnable by some BFL, is also e-provable by the axiom system $\Omega$.

We conjecture that our axiom system {*traditional logic*, *t-axiom*, *e-axiom*, *c-axiom*} is practically complete. Theorems 22 and 23 (compared with Theorems 12 and 13) in a sense support our conjecture. Our conjecture is supported also by the following theorem:

**Theorem 24.** *If $A$ is e-provable formula and $B$ is e-provable formula, then $A \wedge B$ is e-provable formula and $A \vee B$ is e-provable formula.*

We omit here the proof of this theorem, it is not complicated. Let us mention only that an analog of this theorem for $\neg A$ and $A \Rightarrow B$ is not valid.

We conclude with an open problem the solution of which would clarify much (if not everything) for the solution of our conjecture.

**Open Problem.** Let two formulas $A$ and $B$ be provably equivalent in the classical logic. Let $A$ be e-provable in our axiom system. Is the other formula $B$ also e-provable in our axiom system?

## 10. Post scriptum

Carl Smith has been our co-author in 33 published papers. Carl was a very special co-author. He not only produced theorems but also improved our English and taught the psychology of publishing papers in Western scientific journals.

However, this paper is not merely one of jointly written papers.We started the research represented in this paper about eight years ago. Carl participated actively, traveled between Maryland and Latvia many times, produced many versions of axioms, the text of the paper-to-be, examples and counterexamples. His illness was one of the factors why this project was delayed. Now after his untimely death we feel obliged to fulfill Carl's dream and bring the paper to the readers. We miss our friend very much.

## References

[1] D. Angluin, C.H. Smith, Inductive inference: Theory and methods, Computing Surveys 15 (1983) 237–269.

[2] J. Bārzdiņš, R. Freivalds, C. Smith, Learning with confidence, in: C. Puech, R. Reischuk (Eds.), Proceedings of the 13th Symposium on the Theoretical Aspects of Computer Science, in: Lecture Notes in Comput. Sci., vol. 1046, Springer, 1996, pp. 207–218.

[3] J. Bārzdiņš, R. Freivalds, C. Smith, Learning formulae from elementary facts, in: Proceedings of EuroCOLT '97, in: Lecture Notes in Artificial Intelligence, vol. 1208, Springer, 1997, pp. 272–285.
[4] J. Bārzdiņš, R. Freivalds, C. Smith, A logic of discovery, in: Proceedings of Discovery Science '98, in: Lecture Notes in Artificial Intelligence, vol. 1532, Springer, 1998, pp. 401–402.
[5] J. Bārzdiņš, R. Freivalds, C. Smith, Towards a logic of discovery, in: R. Bonner, R. Freivalds (Eds.), Proceedings of the International Workshop on Quantum Computation and Learning, 2000, pp. 110–120.
[6] R. Freivalds, K.M. Podnieks, On computation in the limit by non deterministic Turing machines, in: Theory of Algorithms and Programs, vol. 1, Latvian State University, Riga, USSR, 1974, pp. 25–31 (in Russian).
[7] R. Freivalds, Functions and functionals computable in the limit, in: J. Bārzdiņš (Ed.), Theory of Algorithms and Programs, vol. 1, Latvian State University, Riga, USSR, 1974, pp. 6–19.
[8] R. Freivalds, Minimal Gödel numbers and their identification in the limit, in: Proceedings International Conference on Mathematical Foundations of Computer Science, in: Lecture Notes in Comput. Sci., vol. 32, Springer, 1975, pp. 219–225.
[9] R. Freivalds, Inductive inference of minimal size programs, in: M. Fulk, J. Case (Eds.), Proceedings of the third Annual Workshop on Computational Learning Theory, Morgan Kaufman, 1990, pp. 1–20.
[10] R. Freivalds, Inductive inference of recursive functions: Qualitative theory, in: J. Bārzdiņš, D. Bjorner (Eds.), Baltic Computer Science, in: Lecture Notes in Comput. Sci., vol. 502, Springer, 1991, pp. 77–110.
[11] R. Freivalds, J. Bārzdiņš, K. Podnieks, Inductive inference of recursive functions: Complexity bounds, in: J. Bārzdiņš, D. Bjorner (Eds.), Baltic Computer Science, in: Lecture Notes in Comput. Sci., vol. 502, Springer, 1991, pp. 111–155.
[12] R. Friedberg, Three theorems on recursive enumeration, J. Symbolic Logic 23 (1958) 309–316.
[13] K. Gödel, On undecidable propositions of formal mathematical systems, in: M. Davis (Ed.), The Undecidable, Raven Press, Hewlett, NY, 1965, pp. 39–73.
[14] E.M. Gold, Limiting recursion, J. Symbolic Logic 30 (1965) 28–48.
[15] E.M. Gold, Language identification in the limit, Inform. Control 10 (1967) 447–474.
[16] David Hilbert, Grundlagen der Geometrie (Festschrift zur Feier der Enthüllung des Gauss-Weber-Denkmals in Göttingen), Teubner, Leipzig, 1879.
[17] S.C. Kleene, Introduction to Metamathematics, American Elsevier, 1952.
[18] E. Mendelson, Introduction to Mathematical Logic, third ed., Wadsworth and Brooks/Cole, 1987.
[19] D. Osherson, M. Stob, S. Weinstein, A universal inductive inference machine, J. Symbolic Logic 56 (2) (1991) 661–672.
[20] Giuseppe Peano, Selected works of Giuseppe Peano, with a biographical sketch and bibliography by H.C. Kennedy, London, 1973.
[21] H. Rogers Jr., Theory of Recursive Functions and Effective Computability, McGraw–Hill, New York, 1967.
[22] C.H. Smith, A Recursive Introduction to the Theory of Computation, Springer, 1994.
[23] E. Zermelo, Über Grenzzahlen und Mengenbereiche, Fund. Math. 16 (1930) 29–47.

## Further reading

[24] W. Gasarch, A. Lee, Inferring answers to queries, in: Proceedings of the 10th Annual Conference on Computational Learning Theory, ACM, 1997, pp. 275–284.
[25] D. König, Sur les correspondances multivoques des ensembles, Fund. Math. 8 (1926) 114–134.
[26] Y. Mukouchi, S. Arikawa, Towards a mathematical theory of machine discovery from facts, Theoret. Comput. Sci. 137 (1995) 53–84.
[27] M. Machtey, P. Young, An Introduction to the General Theory of Algorithms, North-Holland, New York, 1978.
[28] R.I. Soare, Recursively Enumerable Sets and Degrees, Springer, New York, 1987.
[29] R. Wiehagen, Limes-Erkennung rekursiver Funktionen durch spezielle Strategien, Elektronische Informationsverarbeitung und Kybernetik 12 (1976) 93–99.